# High-Performance Component Technology

CASC Center for Applied Scientific Computing

## Mission

We are developing software component technology for high-performance scientific computing to address problems of complexity, re-use, and interoperability for laboratory simulation software. Our research at the Center for Applied Scientific Computing (CASC) at the Lawrence Livermore National Laboratory (LLNL) focuses on the unique requirements of scientific computing on parallel machines, such as fast in-process connections among components, language interoperability for scientific languages, and data distribution support for massively parallel components.

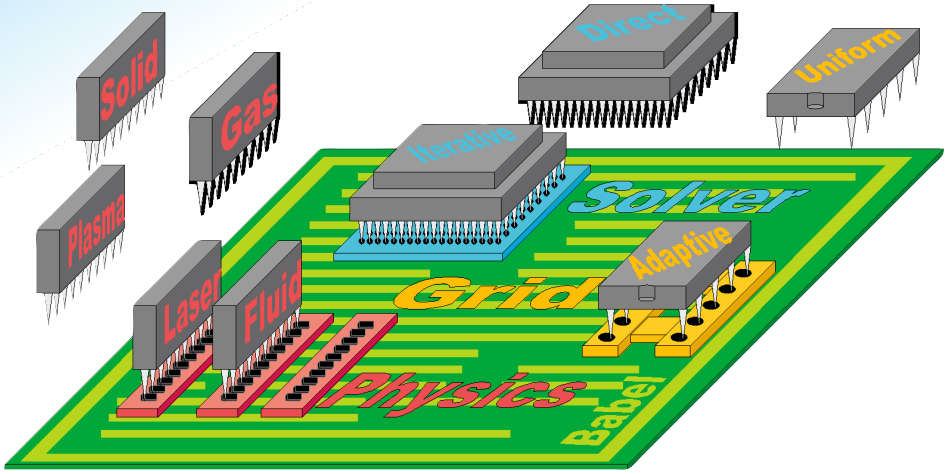


*Figure 1: Component applications are built from component building blocks; components can be thought of as software "integrated circuits."*

Numerical simulations play a vital role in the DOE's science mission as a basic research tool for understanding fundamental physical processes. As simulations become increasingly sophisticated and complex, no single person—or even a single laboratory—can develop scientific software in isolation. Instead, physicists, chemists, mathematicians, and computer scientists concentrate on developing software in their domain of expertise. Computational scientists create simulations by combining these individual software pieces.

It is often difficult to share sophisticated software packages among applications due to differences in implementation languages, programming style, or calling interfaces. In the industrial sector, this problem is solved through the use of component technology. Unfortunately, industry component solutions are inappropriate for parallel scientific computing because they do not support the concept of a "parallel component" that is required for high-performance scientific computing.

## Scope of Project

We are investigating and developing component technology in three primary areas. First, we have developed a software tool called Babel that enables language interoperability among a variety of scientific programming languages, including Fortran, C, C++, Java, and Python. Second, we have developed a web-based component repository called Alexandria for deploying software components. Finally, we are investigating parallel data redistribution issues for communication among distributed components running on differing numbers of parallel processors.

Our Babel tool addresses language interoperability issues for high-performance parallel scientific software. Its purpose is to enable the creation, description, and distribution of language independent software libraries. Babel uses Interface Definition Language (IDL) techniques. An IDL describes the calling interface (but not the implementation) of a particular software library. We have designed a Scientific Interface Definition Language (SIDL) that addresses the unique needs of parallel scientific computing. SIDL supports complex numbers and dynamic multidimensional arrays as well as parallel communication directives that are required for parallel distributed components. As shown in Figure 2, Babel uses this SIDL interface description to generate glue code that allows a software library implemented in one supported language to be called seamlessly from any other supported language. Babel's capability to freely mix languages allows libraries to be written in any supported language and called by any other supported language. For example, an application written in Python could call a solver library written in C, physics packages written in Fortran or C++, and a visualization package written in Java.

Babel currently supports Fortran 77, C, C++, Python, and Java. We plan to add support for Fortran 90 in the near future. We are also researching extensions to SIDL that would add semantic descriptions for the behavior of scientific components.

Another important achievement has been the development of Alexandria, a prototype web-based repository, to



UCRL-TB-145880

encourage the distribution and re-use of scientific software components and libraries. By providing a convenient web-based delivery system, Alexandria lowers the barrier to adopting component technology. We will work with the DOE Common Component Architecture forum (see below) to establish common schema for accessing Alexandria from component tools developed by other DOE collaborators.

Finally, we are investigating parallel data redistribution among components running on different parallel machines and on different numbers of processors. Communication between two applications running on differing numbers of processors requires data redistribution. For example, a simulation code running on thousands of processors may need to communicate mesh data to a visualization server with a relatively small number of processors. The data redistribution for simple data types, such as multidimensional arrays, has been addressed by others in the past. We are investigating general techniques for the sophisticated data structures typically found in complex scientific applications, such as meshes and sparse matrices.

## Common Component Architecture Forum

We are working closely with members of the Common Component Architecture (CCA) forum (see http://www.cca-forum.org). The CCA is a working group of physicists, mathematicians, and computer scientists developing component technology standards that address the high-performance computing needs of the DOE. CCA members include participants from the DOE (ANL, LANL, LBNL, LLNL, ORNL, and SNL) and academia (Indiana University and University of Utah). The CCA is developing a reference implementation of a component technology infrastructure for high-performance computing.

CASC component technology plays an important role in the emerging CCA software infrastructure. Library developers will use SIDL to describe CCA component interfaces, and our Babel tool will provide the underlying language interoperability support for CCA frameworks. Alexandria will store CCA components, library interface descriptions in SIDL, and other component meta-data, such as documentation. We are

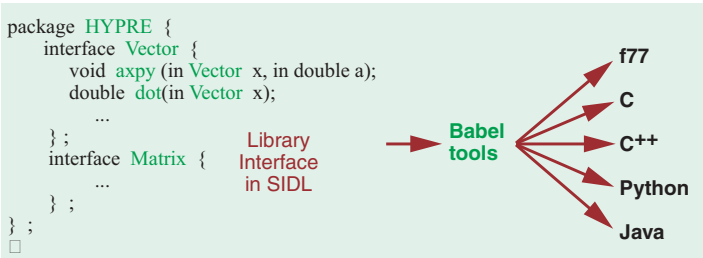


***Figure 2: Babel provides language interoperability using SIDL interface descriptions***

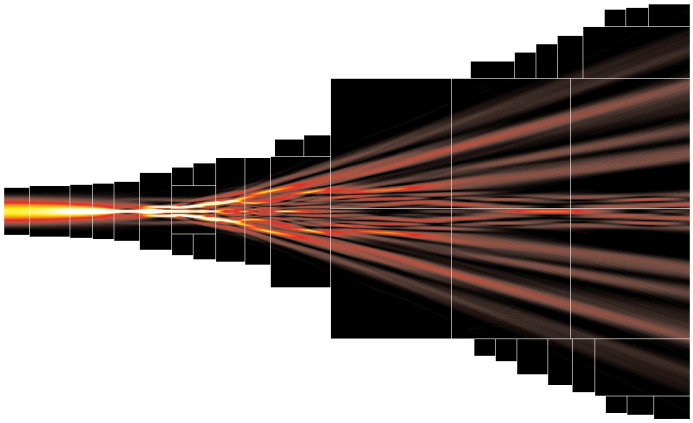

***Figure 3: We are integrating component technology into ALPS to improve interactivity and data analysis capabilities.***

also contributing to the community effort investigating various parallel data redistribution approaches for distributed parallel components.

## Technology Demonstrations

In addition to the CCA, we are collaborating with laboratory research groups to demonstrate component technology in scientific libraries and applications. These collaborations help us understand the issues involved in using advanced software technologies for scientific simulations, and they demonstrate the applicability of component approaches.

In particular, we are working with the *hypre* Scalable Linear Solvers team to integrate Babel language interoperability technology into their solver library. Our technology will enable the *hypre* library, developed using object-oriented techniques in C, to be called from scientific applications written in Fortran 77, C, C++, Java, and Python. We are also working with members of the Advanced Laser Plasma Simulator (ALPS) and the Structured Adaptive Mesh Refinement Application Infrastructure (SAMRAI) projects to use Babel technology to develop an advanced scripting layer for the ALPS framework. This scripting layer will allow computational scientists to more easily analyze ALPS simulation results and compare against other existing computational tools.

## Contact Information

*Additional information about High-Performance Component Technology at LLNL is available from our web site http://www.llnl.gov/CASC/component. You may also contact the team at components@llnl.gov or Scott Kohn at skohn@llnl.gov or 925-422-4022.*

This work was performed under the auspices of the U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.